09/777512

Fig. 1 PRIOR ART

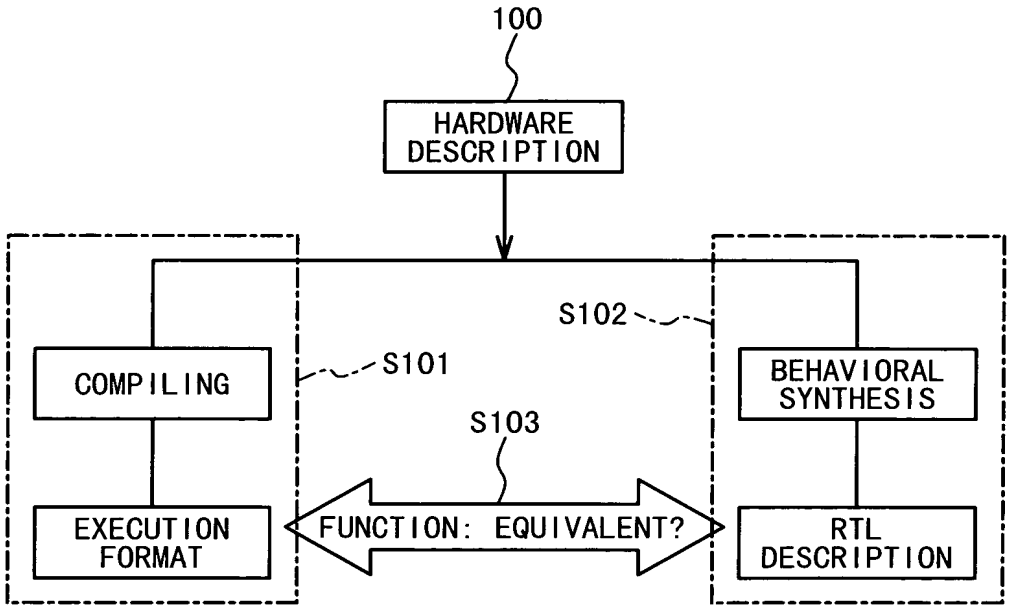


Fig. 2

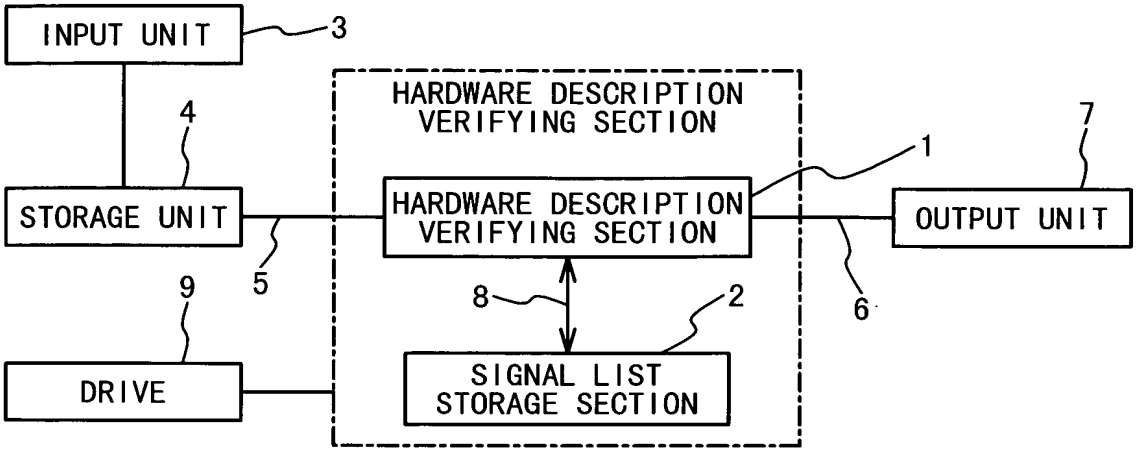


Fig. 3

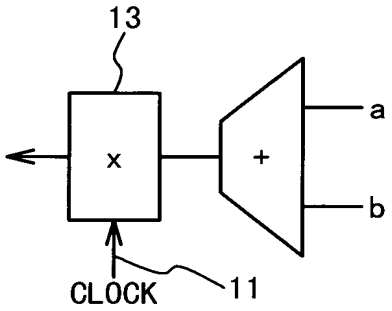


Fig. 4

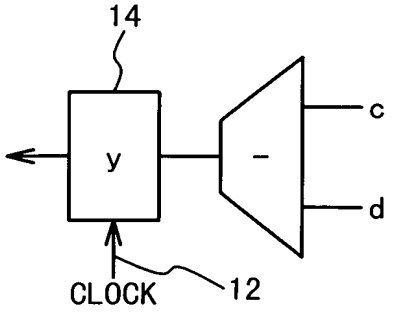


Fig. 5

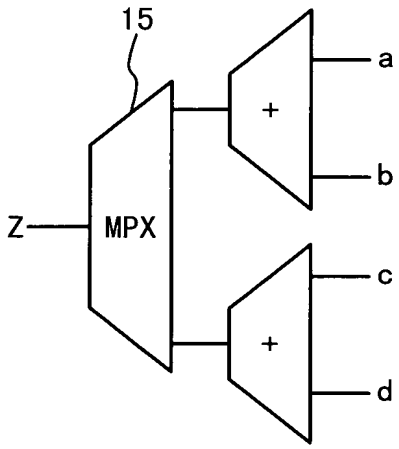


Fig. 6

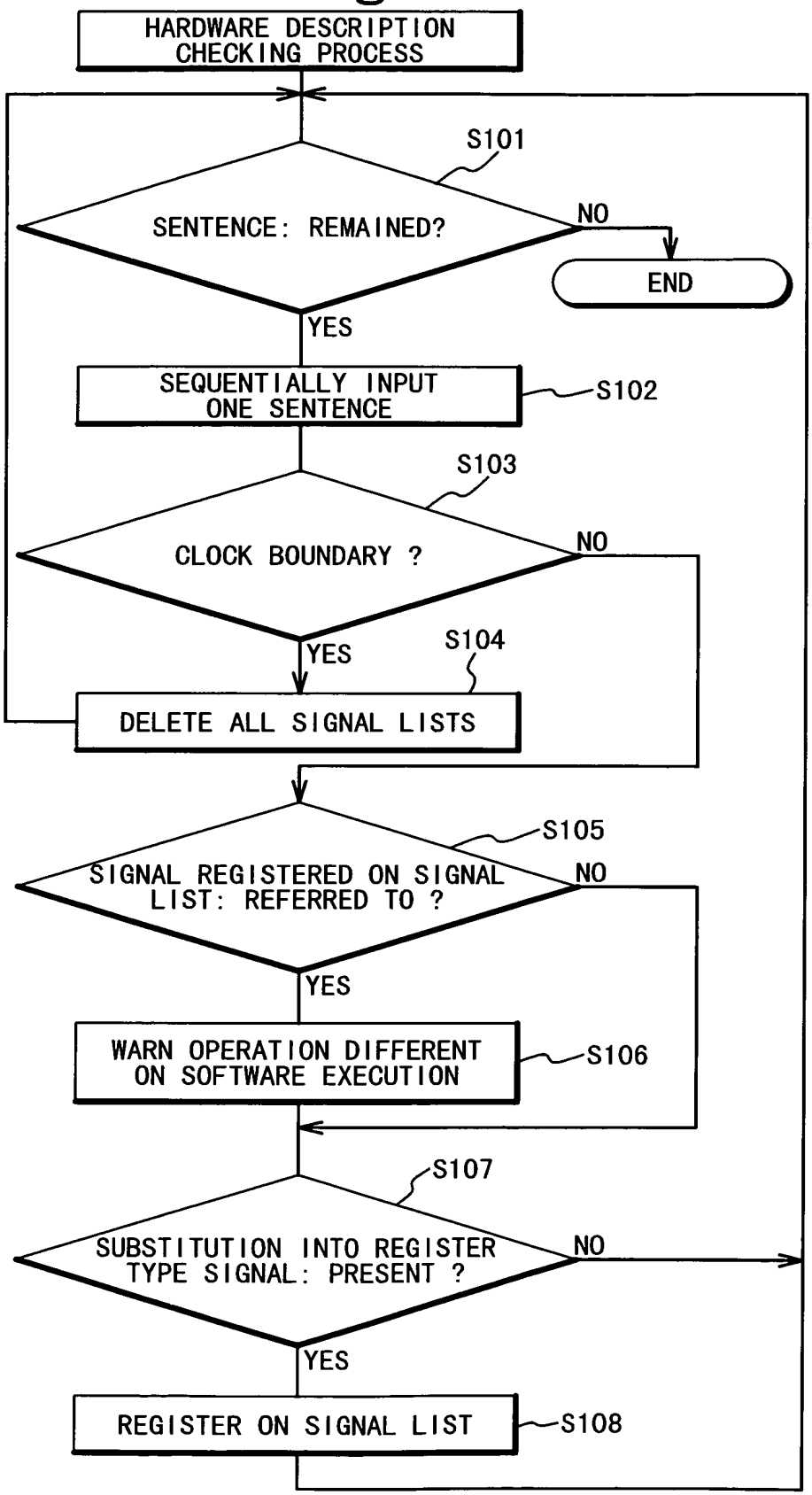

HARDWARE DESCRIPTION CHECKING PROCESS
S101
SENTENCE: REMAINED?
NO
END
YES
SEQUENTIALLY INPUT ONE SENTENCE
S102
S103
CLOCK BOUNDARY ?
NO
YES
S104
DELETE ALL SIGNAL LISTS
S105
SIGNAL REGISTERED ON SIGNAL LIST: REFERRED TO ?
NO
YES
WARN OPERATION DIFFERENT ON SOFTWARE EXECUTION
S106
S107
SUBSTITUTION INTO REGISTER TYPE SIGNAL: PRESENT ?
NO
YES
REGISTER ON SIGNAL LIST
S108

Fig. 7

```
/* REG.C    */

#IFDEF C
# DEFINE REG UNSIGNED INT
#ENDIF

MAIN()
                          /* C */          /* HDL */          /* LIST */
        INT T;
        REG x, y;

        x = 0;            /* x=0 */                          /* {x} */          /* S107 */
        CLOCK();                           /* x=0 */          /* {} */           /* S103 */
        x = 1;            /* x=1 */                          /* {x} */          /* S107 */
        t = 3;            /* t=3 */        /* t=3 */         /* {x} */
        y = x + t;        /* y=4 */                          /* {x,y} */        /* S105,S107 */
        CLOCK();                           /* x=1,Y=3 */      /* {} */           /* S103 */
```

Fig. 8

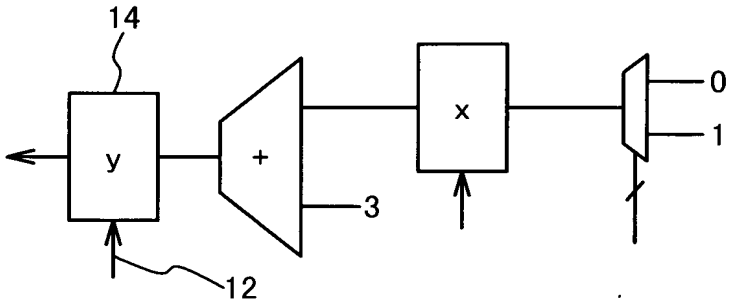


Fig. 9

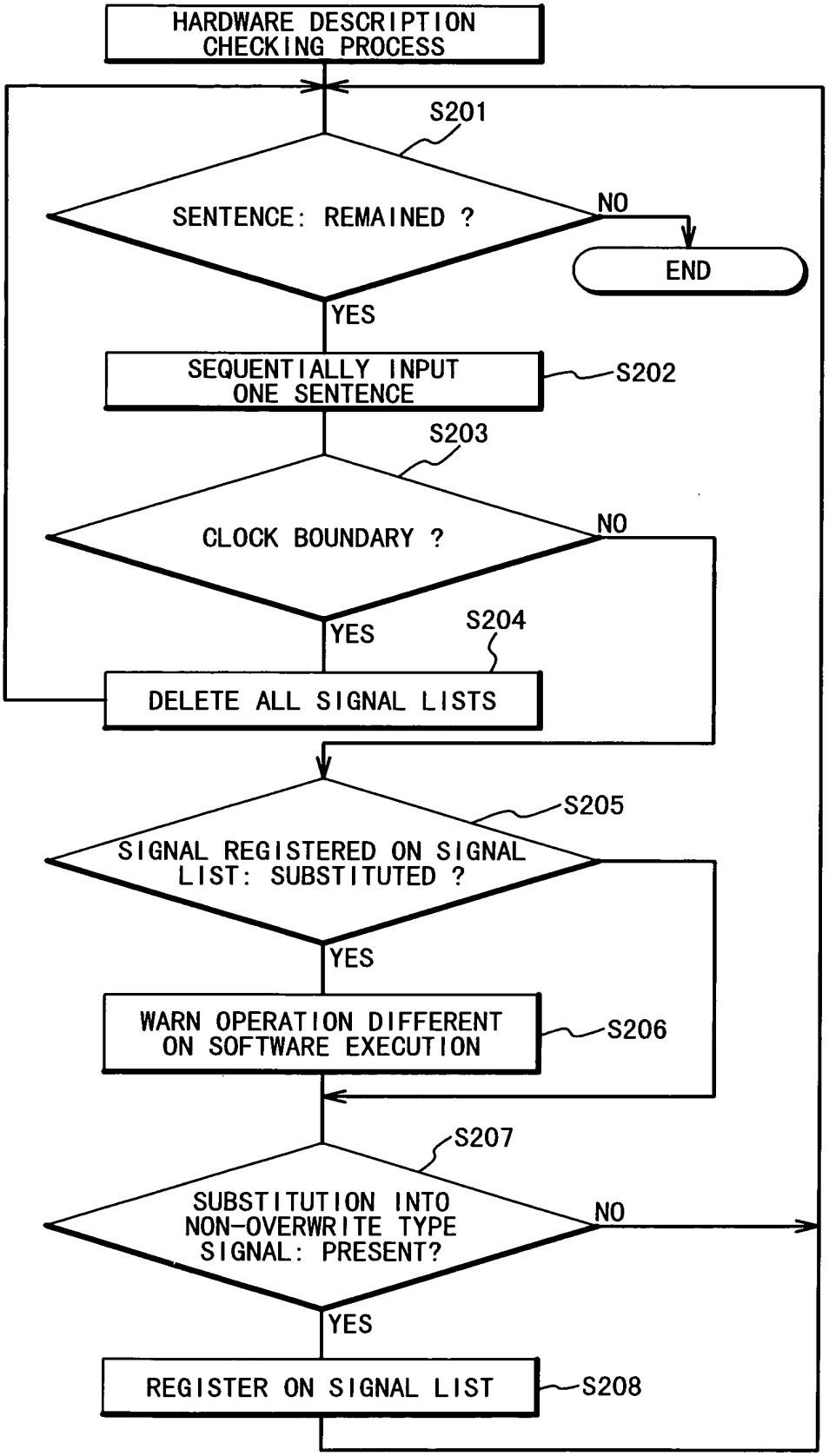

HARDWARE DESCRIPTION CHECKING PROCESS
S201
SENTENCE: REMAINED ?
NO
END
YES
SEQUENTIALLY INPUT ONE SENTENCE
S202
S203
CLOCK BOUNDARY ?
NO
YES
S204
DELETE ALL SIGNAL LISTS
S205
SIGNAL REGISTERED ON SIGNAL LIST: SUBSTITUTED ?
YES
WARN OPERATION DIFFERENT ON SOFTWARE EXECUTION
S206
S207
SUBSTITUTION INTO NON-OVERWRITE TYPE SIGNAL: PRESENT?
NO
YES
REGISTER ON SIGNAL LIST
S208

Fig. 10

```
/* ASSIGN.C */

#IFDEF C
# DEFINE TER UNSIGNED INT
#ENDIF

MAIN()
{                           /* C */          /* HDL */          /* LIST */
        TER z, t;

        z = 0;              /* z=0 */        /* z=0 */          /* {z} */          /* S207 */
        CLOCK();                                                /* {} */           /* S203 */
        z = 1;              /* z=1 */        /* z=? */          /* {z} */          /* S207 */
        t = 3;              /* t=3 */        /* t=3 */          /* {z,t} */        /* S207 */
        z = t + 2;          /* z=5 */        /* z=? */          /* {z,t} */        /* S205,S207 */
        CLCOK();                                                /* {} */           /* S203 */
}
```

Fig. 11

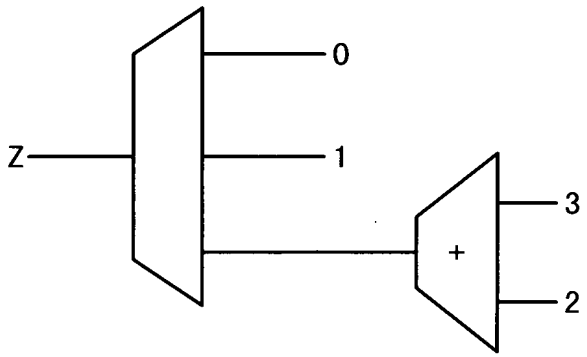


Fig. 12

```
/* TER.C */

#IFDEF C
# DEFINE TER UNSINED INT
#ENDIF

MAIN()
{                               /* C */          /* HDL */          /* LIST */
        TER z, t;

        t = 3;                  /* t=3 */        /* t=3 */          /* {t} */
        CLOCK();                                 /* t=? */          /* {} */
        z = t + 2;              /* z=5 */        /* z=? */          /* {z} */
        CLOCK();                                                    /* {} */
```

Fig. 13

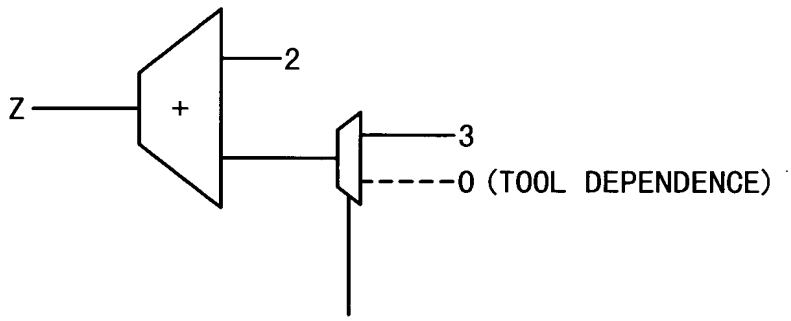


Fig. 14

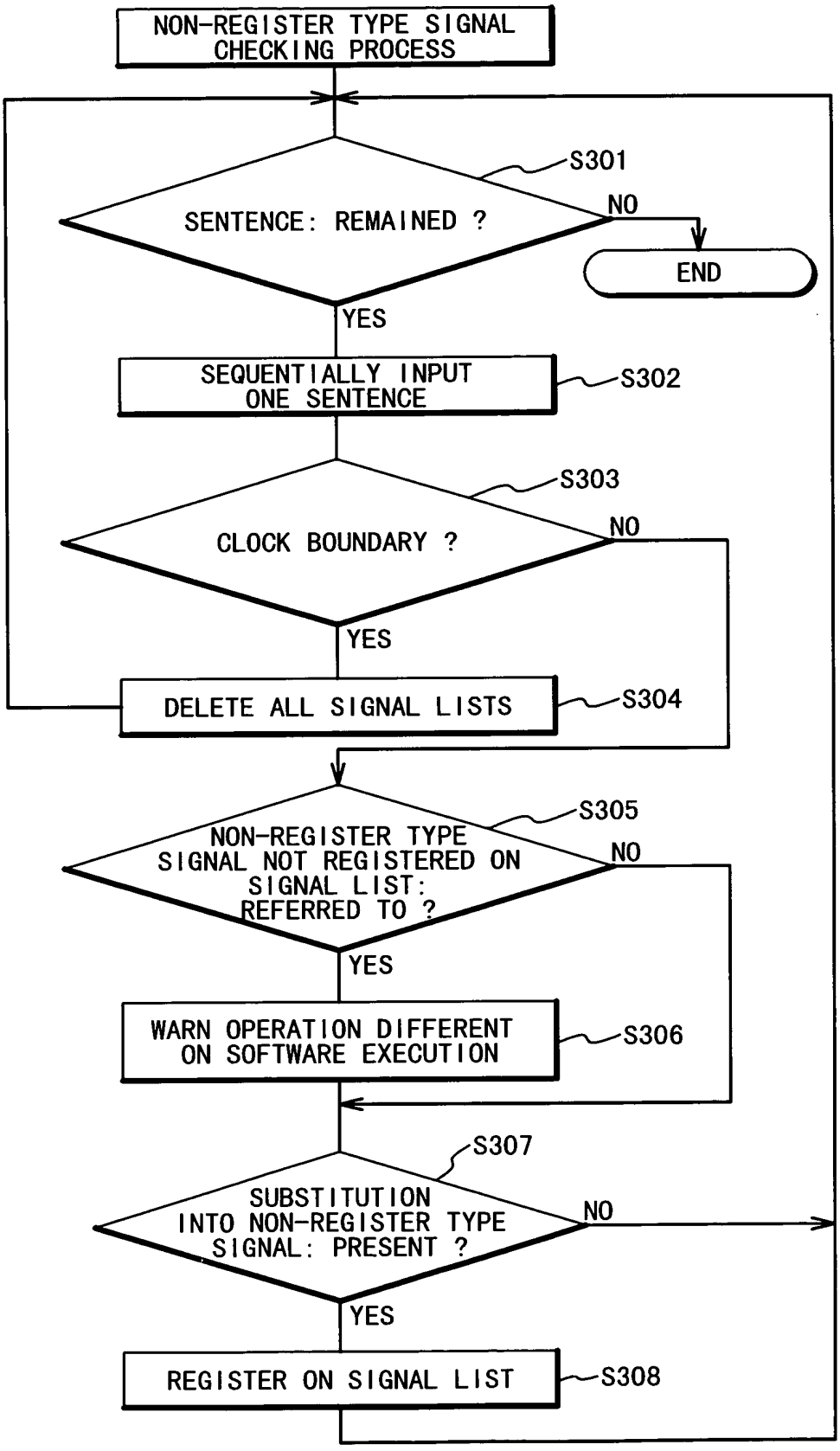

NON-REGISTER TYPE SIGNAL CHECKING PROCESS
S301
SENTENCE: REMAINED ?
NO
END
YES
SEQUENTIALLY INPUT ONE SENTENCE
S302
S303
CLOCK BOUNDARY ?
NO
YES
DELETE ALL SIGNAL LISTS
S304
S305
NON-REGISTER TYPE SIGNAL NOT REGISTERED ON SIGNAL LIST: REFERRED TO ?
NO
YES
WARN OPERATION DIFFERENT ON SOFTWARE EXECUTION
S306
S307
SUBSTITUTION INTO NON-REGISTER TYPE SIGNAL: PRESENT ?
NO
YES
REGISTER ON SIGNAL LIST
S308

Fig. 15

```
/*WIRE.C   */

#IFDEF C
# DEFINE TER UNSIGNED INT
#ENDIF

MAIN()
{                                /* C */          /* HDL */          /* LIST */
        TER z, t;

        t = 3;                   /* t=3 */        /* t=3 */
        CLOCK();                                                     /* {t} */
        z = t + 2;               /* z=5 */        /* z=3 */          /* {} */
        t = 1;                   /* t=1 */        /* t=1 */          /* {z} */
        CLOCK();                                                     /* {z,t} */
                                                                     /* {} */
```

Fig. 16

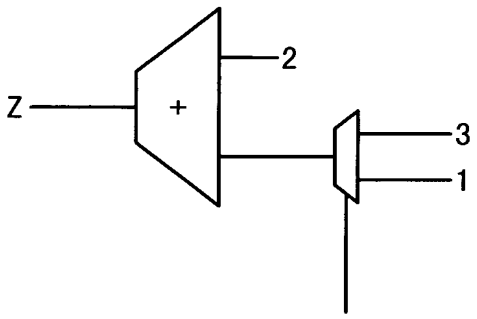


Fig. 17

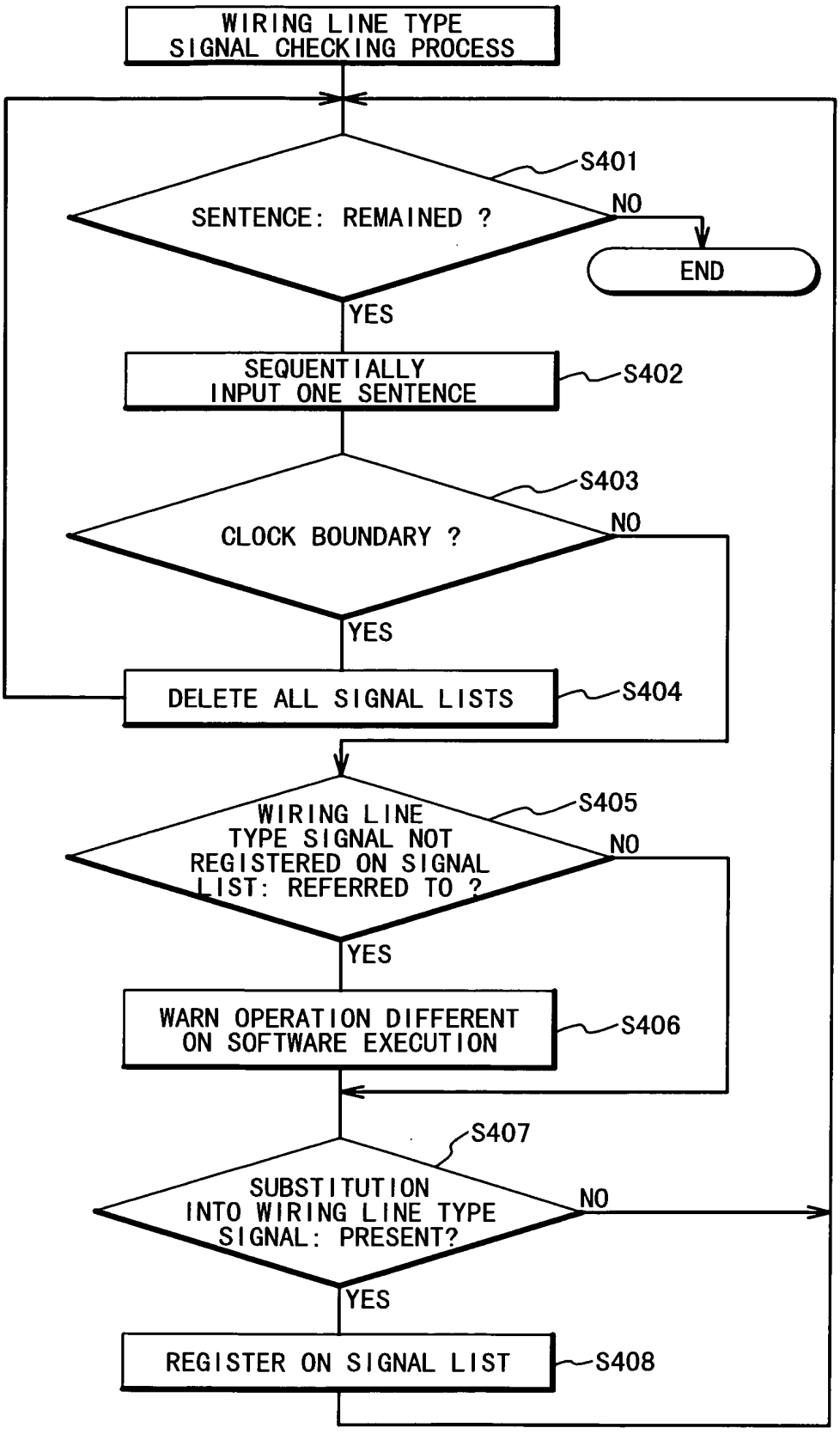

WIRING LINE TYPE SIGNAL CHECKING PROCESS
S401
SENTENCE: REMAINED ?
NO
END
YES
SEQUENTIALLY INPUT ONE SENTENCE
S402
S403
CLOCK BOUNDARY ?
NO
YES
DELETE ALL SIGNAL LISTS
S404
S405
WIRING LINE TYPE SIGNAL NOT REGISTERED ON SIGNAL LIST: REFERRED TO ?
NO
YES
WARN OPERATION DIFFERENT ON SOFTWARE EXECUTION
S406
S407
SUBSTITUTION INTO WIRING LINE TYPE SIGNAL: PRESENT?
NO
YES
REGISTER ON SIGNAL LIST
S408

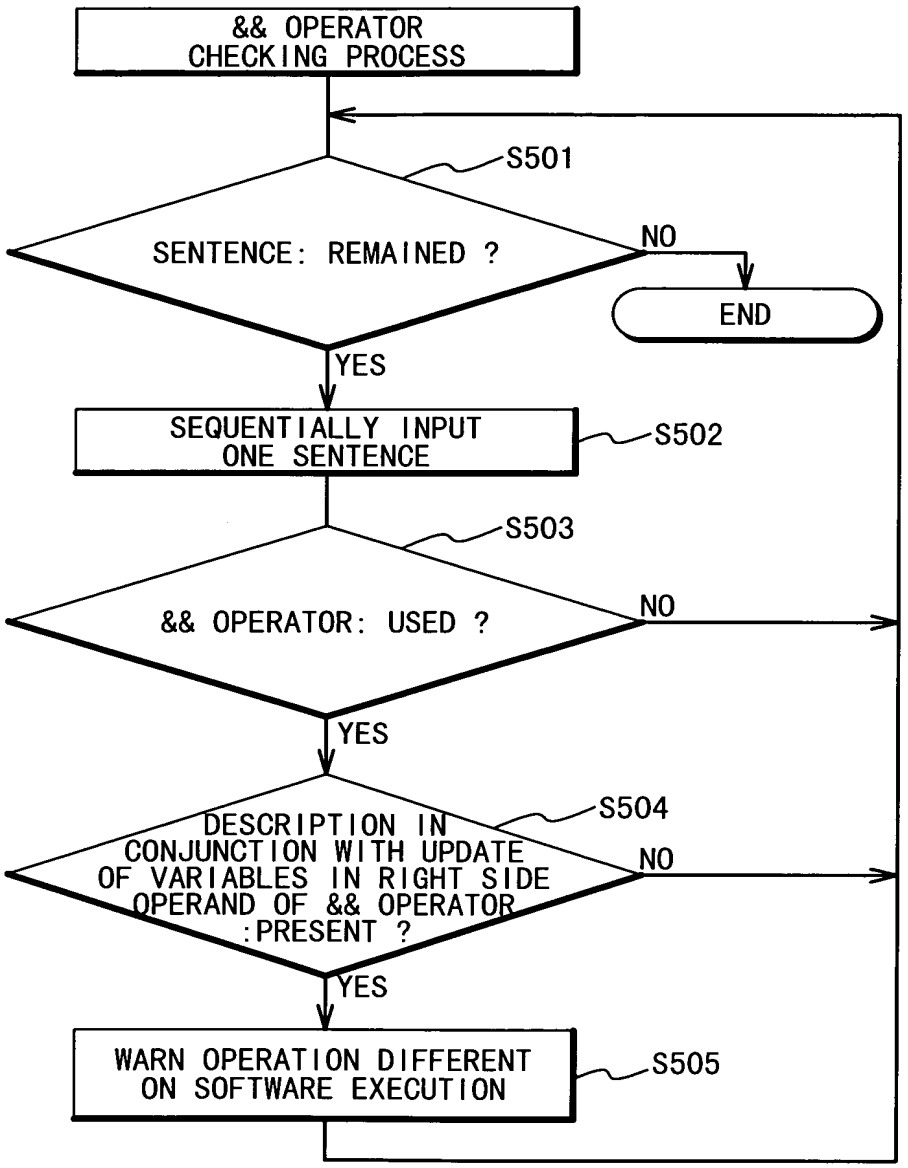
Fig. 18

```
/* AND.C    */

MAIN()
{                                       /* C */          /* HDL */
        INT a,i;

        i = 0;                          /* i=0 */        /* i=0 */
        a = 0;                          /* a=0 */        /* a=0 */
        CLCOK();
        if(i >0 && a++) {               /* a=0 */        /* a=1 */
                i = 0;
        }
        CLCOK();
```

Fig. 19


&& OPERATOR CHECKING PROCESS
S501
SENTENCE: REMAINED ?
NO
END
YES
SEQUENTIALLY INPUT ONE SENTENCE
S502
S503
&& OPERATOR: USED ?
NO
YES
S504
DESCRIPTION IN CONJUNCTION WITH UPDATE OF VARIABLES IN RIGHT SIDE OPERAND OF && OPERATOR :PRESENT ?
NO
YES
WARN OPERATION DIFFERENT ON SOFTWARE EXECUTION
S505